

Apple

\$1.80



Assembly

Line

Volume 6 -- Issue 9

June, 1986

Using the 65816 Stack Relative Mode.	2
Fast 16x16 Multiply & Divide in 65802.	7
The Real Story about DOS and BRUN.	10
Toggling Between Two Values.	13
Using Apple's Protocol Converter	17
Generalized MLI System Error Handling.	24
Practical Application of CRC	30

So Soon?

Another issue of Apple Assembly Line already? Well, readers sent in articles, Bob went on a writing binge, and we've managed to gain over a week in our efforts to get AAL back on schedule. You should all actually receive this issue during the month of June! One side effect of this acceleration is that Bill wasn't ready in time with the code to boot DOS 3.3 from his UniDisk 3.5. It looks like next month for that program and article.

What, Not Yet?

Osborne/McGraw-Hill reports that their copies of 65816 Assembly Language Programming, by Michael Fischer, arrived today (6/3), so our orders should be shipped within two weeks. We'll send them on to our customers just as soon as they arrive. Simon & Schuster has taken over all of Prentice-Hall's titles, so they are now the ones we are bugging about Programming the 65816, by David Eyes. The latest word from S & S is mid-July. Sigh.

We understand that there is a 65816 book from Sybex in the stores, but the people who have seen it aren't very impressed, describing it as a 6502 book with some '816 information gleaned from the data sheets but few examples.

More Disk Utilities

We are now carrying the highly-regarded disk utility package Copy II Plus. This includes disk and file copy programs, catalog and file handling utilities for both DOS and ProDOS, track and sector editing, and much more. List price for all this is only \$39.95, but we'll have it for just \$35 + shipping.

Using the 65816 Stack Relative Mode.....Bob Sander-Cederloir

The 65802 and 65816 have two new address modes that allow you to reach into the stack. The "offset,S" mode lets you access position relative to the stack pointer, and the "(offset,S),Y" mode lets you access data indirectly through an address that is on the stack. The new address modes are available even when the 65802/16 is in the "emulation" mode.

The hardware adds the value of the offset to the current stack pointer to form an effective address. The stack pointer is always pointing one address below the end of the stack. Thus, an address of "1,S" points to the first item on the stack.

These new modes lead to interesting programming possibilities. When you design a subroutine, you have to decide how you are going to pass parameters into and out of the subroutine. Usually we try to use the A, X, and Y registers first. Another method puts the data or the address of the data after the JSR that calls the subroutine. ProDOS MLI calls use this method:

```
JSR $BF00
.DA #$C1,PARMS
```

In another method you push data or data addresses on the stack, and then call the subroutine. This is the preferred method in some computers, but not the 6502. The new modes make this mode work nicely in the 65802/16, though.

I coded up two examples to show how you can use the new modes, both message printing subroutines. The calling method requires telling the subroutine where to find a variable length message. In the first one (lines 1070-1330), I chose to push the address of the text on the stack before calling the printing routine. In the second example (lines 1340-1640), I used the method of storing the message text immediately after the JSR instruction.

Lines 1070-1110 print out two messages, using the first technique. I use the PEA (Push Effective Address) instruction to put the address of the first byte of the message text on the stack. This instruction pushes first the high byte, then the low byte, of the value of the operand. (I think I would prefer to have called it "PSH #value", because that is the effect. Then the PEI opcode, which pushes two bytes from the direct page, could be "PSH zp". But, nobody asked me.)

Anyway, let's look at the PRINT.IT subroutine. When the subroutine starts looking at the stack, it looks like this:

	msg addr lo		4,S

	msg addr hi		3,S

	ret addr lo		2,S

	ret addr hi		1,S

			<---Stack Pointer

S-C Macro Assembler Version 2.0.....DOS \$100, ProDOS \$100, both for \$120
 Version 2.0 DOS Upgrade Kit for 1.0/1.1/1.2 owners.....\$20
 ProDOS Upgrade Kit for Version 2.0 DOS owners.....\$30
 Source Code of S-C Macro 2.0 (DOS only).....additional \$100
 Full Screen Editor for S-C Macro (with complete source code).....\$49
 S-C Cross Reference Utility.....without source code \$20, with source \$50
 RAK-Ware DISASM.....without source code \$30, with source \$50
 S-C Word Processor (with complete source code).....\$50
 DPl8 Source and Object.....\$50
 Double Precision Floating Point for Applesoft (with source code).....\$50
 ES-CAPE (Extended S-C Applesoft Program Editor).....
 Including Version 2.0 With Source Code.....\$50
 ES-CAPE Version 2.0 and Source Code Update (for Registered Owners)....\$30
 Bag of Tricks 2 (Quality Software).....(\$49.95) \$45 *
 Copy II Plus (Central Point Software).....(\$39.95) \$35 *
 Applesoft Toolbox Series (Roger Wagner Publishing).....each (\$39.95) \$36 *
 all four (\$159.80) \$140 *
 MacASM -- Macro Assembler for Macintosh (Mainstay).....(\$150.00) \$100 *
 S-C Documentor (complete commented source code of Applesoft ROMs).....\$50
 Cross Assemblers for owners of S-C Macro Assembler.....\$32.50 to \$50 each
 (Available: 6800/1/2, 6301, 6805, 6809, 68000, Z-80, Z-8, 8048,
 8051, 8085, 1802/4/5, PDP-11, GI1650/70, others)

AAL Quarterly Disks.....each \$15, or any four for \$45
 Each disk contains the source code from three issues of AAL,
 saving you lots of typing and testing.
 The quarters are Jan-Mar, Apr-Jun, Jul-Sep, and Oct-Dec.
 (All source code is formatted for S-C Macro Assembler. Other assemblers
 require some effort to convert file type and edit directives.)

Diskettes (with hub rings)..... package of 20 for \$20 *
 Vinyl disk pages, 6"x8.5", hold two disks each.....10 for \$6 *
 Diskette Mailing Protectors (hold 1 or 2 disks).....40 cents each
 (Cardboard folders designed to fit 6"x9" Envelopes.) or \$25 per 100 *
 Envelopes for Diskette Mailers..... 6 cents each

65802 Microprocessor (Western Design Center).....(\$95) \$50 *
 quikLoader EPROM System (SCRG).....(\$179) \$170 *
 PROMGRAMMER (SCRG).....(\$149.50) \$140 *
 Switch-a-Slot (SCRG).....(\$179.50) \$170 *

"65816/65802 Assembly Language Programming", Fischer.....(\$19.95) \$18 *
 "Programming the 65816", Eyes.....(\$22.95) \$21 *
 "Apple //e Reference Manual", Apple Computer.....(\$24.95) \$23 *
 "Apple //c Reference Manual", Apple Computer.....(\$24.95) \$23 *
 "ProDOS Technical Reference Manual", Apple Computer.....(\$29.95) \$27 *
 "Now That You Know Apple Assembly Language...", Gilder.....(\$19.95) \$18 *
 "Apple ProDOS: Advanced Features for Programmers", Little.....(\$17.95) \$17 *
 "Inside the Apple //c", Little.....(\$19.95) \$18 *
 "Inside the Apple //e", Little.....(\$19.95) \$18 *
 "Apple II+/IIE Troubleshooting & Repair Guide", Brenner.....(\$19.95) \$18 *
 "Apple II Circuit Description", Gayler.....(\$22.95) \$21 *
 "Understanding the Apple II", Sather.....(\$22.95) \$21 *
 "Understanding the Apple //e", Sather.....(\$24.95) \$23 *
 "Enhancing Your Apple II, vol. 1", Lancaster.....(\$15.95) \$15 *
 "Enhancing Your Apple II, vol. 2", Lancaster.....(\$17.95) \$17 *
 "Assembly Cookbook for the Apple II/IIE", Lancaster.....(\$21.95) \$20 *
 "Beneath Apple DOS", Worth & Lechner.....(\$19.95) \$18 *
 "Beneath Apple ProDOS", Worth & Lechner.....(\$19.95) \$18 *
 "Real Time Programming -- Neglected Topics", Foster.....(\$9.95) \$9 *
 "Microcomputer Graphics", Myers.....(\$14.95) \$14 *
 "Assem. Language for Applesoft Programmers", Finley & Myers.....(\$18.95) \$18 *
 "Assembly Lines -- the Book", Wagner.....(\$19.95) \$18 *
 "AppleVisions", Bishop & Grossberger.....(\$39.95) \$36 *

* On these items add \$2.00 for the first item and
 \$.75 for each additional item for US shipping.

Foreign customers inquire for postage needed.

Texas residents please add 6 1/8 % sales tax to all orders.

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
 *** (214) 324-2050 ***
 *** Master Card, VISA, Discover and American Express ***

The LDA (3,S),Y instruction at line 1240 takes the address at 3,S and 4,S (which is the address of the first byte of the message) and adds the Y-register to it; then the LDA opcode picks up the message byte. After printing all the message and finding the terminating 00 byte, lines 1290-1320 move the return address up two slots higher in the stack (over the top of the message address). At the same time, the original copy of the return address is removed from the stack. Then a simple RTS takes us back to the caller, with a clean stack.

The second example uses a "message buried in the code" method. When PRINT.MSG looks at the stack, only the return address is there. The return address points to the third byte of the JSR instruction, one byte before the message text. Therefore the printing loop in lines 1500-1550 starts with Y=1. Lines 1560-1620 add the message length to the return address, so that an RTS opcode will return to the caller just past the message.

```

1000 *SAVE S.TEST 65816 CALLING SEQUENCES
1010 *-----
1020 .OP 65816
1030 *-----
1040 * PEA address of message text
1050 * JSR PRINT.IT
1060 *-----
000800- F4 0D 08 1070 T1 PEA MESSAGE.1
000803- 20 29 08 1080 JSR PRINT.IT
000806- F4 1B 08 1090 PEA MESSAGE.2
000809- 20 29 08 1100 JSR PRINT.IT
00080C- 60 1110 RTS
1120 *-----
1130 MESSAGE.1
1140 .HS 8D
00080D- 8D
00080E- CD C5 D3 D3
000812- C1 C7 C5 A0
000816- CF CE C5
000819- 8D 00 1150 .AS -/MESSAGE ONE/
1160 .HS 8D.00
1170 MESSAGE.2
1180 .HS 8D
00081B- 8D
00081C- CD C5 D3 D3
000820- C1 C7 C5 A0
000824- D4 D7 CF
000827- 8D 00 1190 .AS -/MESSAGE TWO/
1200 .HS 8D.00
1210 *-----
1220 PRINT.IT
1230 LDY #0 STARTING INDEX
00082B- B3 03 1240 .1 LDA (3,S),Y NEXT CHARACTER OF MESSAGE
00082D- F0 06 1250 BEQ .2 ...TERMINATING $00
00082F- 20 ED FD 1260 JSR $FDED PRINT THE CHAR
000832- C8 1270 INY
000833- D0 F6 1280 BNE .1 ... ALWAYS
000835- 68 1290 .2 PLA MOVE RETURN ADDRESS
000836- 83 02 1300 STA 2,S OVER THE TOP OF THE
000838- 68 1310 PLA MESSAGE ADDRESS, PRUNING
000839- 83 02 1320 STA 2,S THE STACK
00083B- 60 1330 RTS
1340 *-----
1350 * JSR PRINT.MSG
1360 * text of message, terminating zero
1370 *-----
1380 T2
00083C- 20 69 08 1390 JSR PRINT.MSG
00083F- 8D 1400 .HS 8D
000840- CD C5 D3 D3
000844- C1 C7 C5 A0
000848- C1 C6 D4 C5
00084C- D2 A0 CA D3
000850- D2 1410 .AS -/MESSAGE AFTER JSR/
000851- 8D 00 1420 .HS 8D.00
000853- 20 69 08 1430 JSR PRINT.MSG
000856- 8D 1440 .HS 8D

```

```

000857- C1 CE CF D4
000858- C8 C5 D2 A0
00085F- CD C5 D3 D3
000863- C1 C7 C5
000866- 8D 00
000868- 60
1450 .AS -/ANOTHER MESSAGE/
1460 .HS 8D.00
1470 RTS
1480 #-----
1490 PRINT.MSG
000869- A0 01 1500 LDY #1 POINT TO FIRST CHAR
00086B- B3 01 1510 .1 LDA (1,S),Y GET NEXT CHAR
00086D- F0 06 1520 BEQ .2 ...TERMINATING $00
00086F- 20 ED FD 1530 JSR $FDED PRINT THE CHAR
000872- C8 1540 INY
000873- D0 F6 1550 BNE .1 ...ALWAYS
000875- 98 1560 .2 TYA ADJUST THE RETURN ADDRESS
000876- 18 1570 CLC BY ADDING THE MESSAGE LENGTH
000877- 63 01 1580 ADC 1,S
000879- 83 01 1590 STA 1,S
00087B- A9 00 1600 LDA #0 THE HIGH BYTE TOO
00087D- 63 02 1610 ADC 2,S
00087F- 83 02 1620 STA 2,S
000881- 60 1630 RTS RETURN TO CALLER
1640 #-----

```

It might be instructive to look at how these two examples could be code in a plain 6502 environment. First, we must replace the PEA opcodes in lines 1070 and 1090 with the following:

```

LDA #MESSAGE
PHA
LDA /MESSAGE
PHA

```

Then PRINT.IT would require using temporary memory somewhere or writing self-modifying code. With a pointer in page zero, it could work like this:

```

00- 1250 RETURN.SAVE .EQ $00,01
02- 1260 PNTR .EQ $02,03
1270 PRINT.IT
082F- 68 1280 PLA POP RETURN ADDRESS
0830- 85 01 1290 STA RETURN.SAVE+1
0832- 68 1300 PLA
0833- 85 00 1310 STA RETURN.SAVE
0835- 68 1320 PLA POP MESSAGE ADDRESS
0836- 85 03 1330 STA PNTR+1
0838- 68 1340 PLA
0839- 85 02 1350 STA PNTR
083B- A0 00 1360 LDY #0 STARTING INDEX
083D- B1 02 1370 .1 LDA (PNTR),Y NEXT CHARACTER OF MESSAGE
083F- F0 06 1380 BEQ .2 ...TERMINATING $00
0841- 20 ED FD 1390 JSR $FDED PRINT THE CHAR
0844- C8 1400 INY
0845- D0 F6 1410 BNE .1 ...ALWAYS
0847- A5 00 1420 .2 LDA RETURN.SAVE
0849- 48 1430 PHA RELOAD RETURN ADDRESS
084A- A5 01 1440 LDA RETURN.SAVE+1
084C- 48 1450 PHA
084D- 60 1460 RTS RETURN TO CALLER

```

PRINT.MSG also can be written in pure 6502 code with either self-modifying code or a pointer in page zero. Here is the self-modifying version:

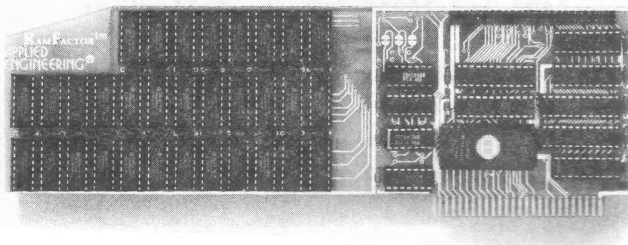
Continued on page 14

RamFactor™

All the Performance, Speed, and Software Compatibility of RamWorks™ in a Slot 1 through 7 Card.

That's right! Now Applied Engineering offers you a choice. While RamWorks is the clear winner for the auxiliary slot in a IIe, RamFactor is the standard for slots 1 through 7. Now anyone with an Apple II+, Franklin, or Apple IIe preferring to use slots 1 through 7 can now enjoy the speed and performance that until now was only available with RamWorks.

With RamFactor, you'll be able to instantly add another 256K, 512K, or a full 1 meg on the main board and up to 16 meg with additional piggyback card. And since virtually all software is automatically compatible with RamFactor, you'll immediately be able to load programs into RamFactor for instantaneous access to information. You'll also be able to store more data for larger word processing documents, bigger data bases, and expanded spreadsheets.



Very Compatible

All the leading software is already compatible with RamFactor. Programs like AppleWorks, Pinpoint, BPI, Managing Your Money, Dollars and Sense, SuperCalc 3A, PFS, MouseWrite, MouseDesk, MouseCalc, Sensible Speller, Applewriter IIe, Business Works, ReportWorks, Catalyst 3.0 and more. And RamFactor is fully ProDos, DOS 3.3, Pascal 1.3 and CP/M compatible. In fact, no other memory card (RamWorks excepted) is more compatible with commercial software.

AppleWorks Power

There are other slot 1-7 cards that give AppleWorks a larger desktop, but that's the end of their story. But RamFactor is the only slot 1-7 card that increases AppleWorks internal memory limits, increasing the maximum number of lines permitted in the word processor, and RamFactor is the only standard slot card that will automatically load AppleWorks into RAM dramatically increasing speed and eliminating the time required to access the program disk, it will even display the time and date on the AppleWorks screen with any ProDos clock. RamFactor will automatically segment large files so they can be saved on 5¼", 3½", and hard disks. All this performance is available to anyone with an Apple IIe or II+ with an 80 column card.

RamFactor, no other standard slot card comes close to enhancing AppleWorks so much.

True 65C816 16 Bit Power

RamFactor has a built-in 65C816 CPU port for direct connection to our IIe 65C816 card for linearly addressing up to 16 meg for the most powerful 16 bit applications (II+ 65C816 card under development.)

Powerful Program Switcher

With RamFactor, you can organize memory into multiple work areas and switch between them. Each work area can contain different programs and even different operating systems. Now you can switch from one program to another or even switch from AppleWorks to DOS 3.3 to CP/M to Pascal to ProDos in under a second. And with our Battery back-up option, you can have permanent storage for up to 20 years.

Quality and Support of the Industry Leader

RamFactor is from Applied Engineering, the largest, most well supported manufacturer of Apple peripherals and the inventor of large RAM cards for the Apple. With our 5 year no hassle warranty and outstanding technical support, you're assured of the most trouble free product you can buy.

Features:

- Up to 16 meg total memory; 256K to 1 meg on main board. Up to 16 meg with additional memory on piggyback card.
- Fully Apple II Memory Expansion compatible
- Compatible with Apple IIe, II+ and Franklin
- Battery back-up option allows you to turn on your Apple and run your favorite programs in less than 1 second!
- Automatically recognized by ProDos, DOS 3.3, Pascal and CP/M
- Built-in RamDrive™ software (a true RAM disk not disk caching)
- Systems are directly bootable from RamFactor if desired
- Built-in linear addressing 16 bit co-processor port
- Built-in self diagnostic software
- Automatic expansion with AppleWorks 1.3 or later
- Allows Apple II+ and IIe to run your AppleWorks without buying additional software
- Accelerates AppleWorks
- Displays time and date on the AppleWorks screen with any ProDos clock
- Fits any I/O slot except slot 3
- Fully socketed and user upgradeable
- Much, much more

RamFactor with 256K	\$239
RamFactor with 512K	\$289
RamFactor with 1 MEG	\$389
RamFactor with 2-16 MEG	CALL
Battery Back-up Option	\$179
65C816 16 Bit Card	\$159

Order RamFactor today... with 15 day money back guarantee and our "no hassle" five year warranty. Call 9 a.m. to 11 p.m. - days, or send check or money order to Applied Engineering. MasterCard, Visa and C.O.D. welcome. Texas residents add 5½% sales tax. Add \$10.00 if outside USA.

AE Applied Engineering
The Apple enhancement experts.

(214) 241-6060

P.O. Box 798, Carrollton, TX 75006

Recently I needed a 16-bit multiplication subroutine in my 65802-enhanced Apple II. Naturally, I needed one that was both fast and short. I referred back to the Jan 86 AAL, which contained several examples for the 65802. The one named FASTER caught my fancy because it seemed a good compromise between size and speed. Then I made some changes which I think significantly improve it.

I noted that when you ROR the low half of the product into the multiplier, you get a bit out. This bit remains in the carry. If the low-product and the multiplier share the same location, then you can ROL in the low-product bit and ROL out the multiplier bit at the same time, instead of loading and LSR-ing the multiplier. By not having to load the multiplier, the Accumulator is free to contain the high half of the product without saving and loading it each time around. The result is rather more compact, fitting into 35 bytes (FASTER took 42 bytes).

It is also faster. By my calculations, the best and worst cases take 335 and 383 cycles, respectively. This includes the JSR to call the subroutine and the RTS to get back.

At the expense of two more bytes, I can save nine more cycles: delete line 1240 and add the following:

```
1304    ROR
1305    ROR A
```

This avoids the 17th trip through the loop, whose only purpose was to roll-in the final bit of the product.

By the way, some assemblers use the syntax "ROR A" to rotate the contents of the A-register. The S-C Macro Assembler and some others use the syntax "ROR" with a blank operand field for that mode. Then "ROR A" means to rotate the contents of the variable named "A", as in my program. To avoid confusion, you might want to change the variable names, avoiding the name "A".

```

1000 *SAVE BUTTERILL'S MULTIPLY
1010 *-----
1020 * 16 BIT MULTIPLY FOR 65802
1030 * MULTIPLIES A BY B
1040 * LEAVES ANSWER IN A & B
1050 *-----
00- 1060 A      .EQ 0,1      MULTIPLIER, PRODUCT-LO
02- 1070 B      .EQ 2,3      MULTIPLICAND, PRODUCT-HI
1080 *-----
1090 *   TIMING:  B=$0000 -- 27 CYCLES
1100 *             A=$0000 -- 335 CYCLES
1110 *             A=$FFFF -- 383 CYCLES
1120 *   (INCLUDING JSR AND RTS)
1130 *-----
      1140      .OP 65802
      1150 MULT16
000800- 18      1160      CLC          ENTER FROM 6502
000801- FB      1170      XCE
000802- C2 20   1180      REP #$20
000804- A5 02   1190      LDA B          IF B ZERO,
000806- F0 17   1200      BEQ .90      THEN BY-PASS
000808- C6 02   1210      DEC B

```

```

00080A- A9 00 00      1220      LDA #0000
00080D- A2 10        1230      LDX #16      FOR 16 BITS
00080F- 18           1240      CLC          FOR 17'TH CYCLE
000810- 6A           1250 .10     ROR          ROLL OUT PRODUCT BIT
000811- 66 00        1260      ROR A        ROLL IN 'PLIER BIT
000813- 90 02        1270      BCC .20
000815- 65 02        1280      ADC B
000817- CA           1290 .20     DEX
000818- 10 F6        1300      BPL .10      CYCLES 17 TIMES
00081A- 85 02        1310      STA B
00081C- 38           1320 .30     SEC          EXIT TO 6502
00081D- FB           1330      XCE
00081E- 60           1340      RTS
00081F- 85 00        1350 .90     STA A        PROCEDURE FOR B=0
000821- 80 F9        1360      BRA .30
1370 *-----

```

A 16-bit by 16-bit division seems inherently messier. First, the divisor must be shifted left until it is at least greater than half the dividend. One can do a fast cycle which shifts the divisor all the way to the left, but for every shift left in this loop, the divisor must be shifted right again in the second (subtracting) loop.

In practice, I feel that the values would not be randomly distributed, but would be biased toward smaller values. I'm more likely to divide by 7 than by 32973, for example. Therefore it is worthwhile putting in the extra code to shift left only as far as is necessary. The scaling portion in my subroutine, lines 1240-1300, shift the divisor until either bit 15 = 1 or the divisor equals/exceeds the dividend.

In the second loop, lines 1310-1400, the shifted divisor is repeatedly compared to the dividend. If it is smaller, it is subtracted and a 1-bit goes into the quotient; otherwise a 0-bit goes in. The loop stops after it has operated with the divisor shifted back to its original position. This is ordinary long division, in binary. The comparison-subtraction is performed from one to 16 times, depending on the values.

As I calculate it, the best case (dividend=divisor) takes 82 cycles. The worst case, which I think would be \$FFFF/1, takes 676 cycles. The time is a function of the number of significant bits in the answer.

[John also wrote a nice demonstration driver for his subroutines, allowing you to enter two hexadecimal values and see the result in hexadecimal. The source code for the demo is included on the monthly/quarterly disk.]

```

1000 *SAVE BUTTERILL'S DIVIDE
1010 *-----
1020 * 16 BIT DIVIDE WITH REMAINDER
1030 * DIVIDE B BY A
1040 * LEAVES QUOTIENT IN B,
1050 *      REMAINDER IN A
1060 *-----
1070 *   TIMING:  A=$0000 -- 39 cycles
1080 *             B>$7FFF -- 71 or 74 cycles
1090 *             A=B    -- 82 cycles
1100 *             A=1,B=$FFFF -- 676 cycles
1110 *-----
00- 1120 A      .EQ 0,1      DIVISOR, REMAINDER
02- 1130 B      .EQ 2,3      DIVIDEND, QUOTIENT
1140 *-----

```



```

1150 .OP 65802
1160 DIV16
1170 CLC ENTER FROM 6502
1180 XCE NATIVE MODE
1190 REP #20 A-REG 16 BITS
1200 LDX #0 START SCALE CNTR
1210 LDA A GET DIVISOR
1220 BEQ .90 ...ZERO DIVISOR
1230 BMI .30 ...DIVISOR > $7FFF
1240 *---SCALE DIVISOR---
1250 .10 CMP B ALIGN A TO LEFT
1260 BCS .20 UNTIL > B
1270 INX OR BIT 15 SET
1280 ASL & COUNT IN X
1290 BPL .10
1300 .20 STA A SCALED DIVIDEND
1310 *---START SUBTRACTING---
1320 .30 LDA B GET DIVIDEND
1330 STZ B CLEAR QUOTIENT
1340 .40 CMP A REPEATED CONDITIONAL
1350 BCC .50 SUBTRACTION.
1360 SBC A
1370 .50 ROL B ROL IN 1 IF SUBT.
1380 LSR A 0 IF NO SUBT.
1390 DEX
1400 BPL .40
1410 STA A REMAINDER
1420 *---RETURN TO CALLER---
1430 .60 SEC EXIT TO 6502
1440 XCE
1450 RTS
1460 *---FOR X/O, GIVE 0,0 ANSWER---
1470 .90 STA B DIVISION BY ZERO
1480 BRA .60
1490 *-----

```

```

000800- 18
000801- FB
000802- C2 20
000804- A2 00
000806- A5 00
000808- F0 22
00080A- 30 0A

```

```

00080C- C5 02
00080E- B0 04
000810- E8
000811- 0A
000812- 10 F8
000814- 85 00

```

```

000816- A5 02
000818- 64 02
00081A- C5 00
00081C- 90 02
00081E- E5 00
000820- 26 02
000822- 46 00
000824- CA
000825- 10 F3
000827- 85 00

```

```

000829- 38
00082A- FB
00082B- 60

```

```

00082C- 85 02
00082E- 80 F9

```

We Make Measurement And Control Easy!

12 BIT, 16 CHANNEL PROGRAMMABLE GAIN A/D

- All new 1984 design incorporates the latest in state-of-art I.C. technologies.
- Complete 12 bit A/D converter, with an accuracy of 0.02%!
- 16 single ended channels (single ended means that your signals are measured against the Apple's GND) or 8 differential channels. Most all the signals you will measure are single ended.
- 9 software programmable full scale ranges, any of the 16 channels can have any range at any time. Under program control, you can select any of the following ranges: ± 10 volts, ± 5 V, ± 2.5 V, ± 1.0 V, ± 500 mV, ± 250 mV, ± 100 mV, ± 50 mV, or ± 25 mV.
- Very fast conversion (25 micro seconds).
- Analog input resistance greater than 1,000,000 ohms.
- Laser-trimmed scaling resistors.
- Low power consumption through the use of CMOS devices.
- The user connector has +12 and -12 volts on it so you can power your sensors.
- Only elementary programming is required to use the A/D.
- The entire system is on one standard size plug in card that fits neatly inside the Apple.
- System includes sample programs on disk.

PRICE \$319

A few applications may include the monitoring of:

- flow
- temperature
- humidity
- wind speed
- wind direction
- light intensity
- pressure
- RPM
- soil moisture and many more.

A/D & D/A

A/D Features:

- Single PC card
- 8 channels A/D
- 8 channels D/A
- Superfast conversion time
- Very easy programming
- Many analog ranges
- Manual contains sample applications

A/D SPECIFICATIONS

- 0.3% accuracy
- On-board memory
- Fast conversion (0.78 MS per channel)
- A/D process totally transparent to Apple (looks like memory)
- User programmable input ranges are 0 to 10 volts, 0 to 5, -5 to +5, -2.5 to +2.5, -5 to 0, -10 to 0.

The A/D process takes place on a continuous, channel sequencing basis. Data is automatically transferred to its proper location in the on-board RAM. No A/D converter could be easier to use.

D/A SPECIFICATIONS

- 0.3% accuracy
 - On-board memory
 - On-board output buffer amps can drive 5 MA
 - D/A process is totally transparent to the Apple (just poke the data)
 - Fast conversion (0.03 MS per channel)
 - User programmable output ranges are 0 to 5 volts and 0 to 10 volts
- The D/A section contains 8 digital to analog converters, with output buffer amplifiers and all interface logic on a single card. On-card latches are provided for each of the eight D/A converters. No D/A converter could be easier to use. The on-board amplifiers are laser-trimmed during manufacture, thereby eliminating any requirement for off-set nulling.

PRICE \$199

SIGNAL CONDITIONER

Our 8 channel signal conditioner is designed for use with both our A/D converters. This board incorporates 8 F.E.T. op-amps, which allow almost any gain or offset. For example, an input signal that varies from 2.00 to 2.15 volts or a signal that varies from 0 to 50 mV can easily be converted to 0-10V output for the A/D.

The signal conditioner's outputs are on a high quality 16 pin gold I.C. socket that mates the one on the A/D's to a simple ribbon cable connects the two. The signal conditioner can be powered by your Apple or from an external supply.

FEATURES

- 4.5" square for standard card cage and 4 mounting holes for standard mounting. The signal conditioner does not plug into the Apple. It can be located up to 1/2 mile away from the A/D.
- 22 pin 156 spacing edge card input connector (extra connectors are easily available i.e. Radio Shack).
- Large bread board area.
- Full detailed schematic included.

PRICE \$79

I/O 32

- Provides 4, 8-Bit programmable I/O Ports
- Any of the 4 ports can be programmed as an input or an output port
- All I/O lines are TTL (0-5 volt) compatible
- Your inputs can be anything from high speed logic to simple switches
- Programming is made very easy by powerful on-board firmware
- The I/O 32 is your best choice for any control application

The I/O manual includes many programs for inputs and outputs.

Some applications include:

Burglar alarm, three lion yowling, use with relays to turn on lights, sound buzzers, start motors, control tape recorders and printers, use with digital joystick.

PRICE \$89

Please see our other full page ad in this magazine for information on Applied Engineering's Timemaster Clock Card and other products for the Apple.

Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products compatible with Apple II and IIe.

Applied Engineering's products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle three year warranty.

Texas Residents Add 5% Sales Tax
Add \$10.00 if Outside U.S.A.

Send Check or Money Order to
APPLIED ENGINEERING
P.O. Box 798
Carrollton, TX 75006

Call (214)241-6060
9 a.m. to 11 p.m. 7 days a week
Mastercard, Visa & C.O.D. Welcome
No extra charge for credit cards

I was wrong. Some of you were kind enough to point it out. John Butterill sent a letter, and others called (sorry, names forgotten). I said, in the January 1986 AAL, that the reason BRUNning programs from inside Applesoft programs often did not work was the fact that DOS used a JMP rather than a JSR to call your program.

The truth is that DOS does call your program with a JMP, but there is still a return address on the stack. The BRUN command processor itself was called with a JSR, in a way. At \$A17A there is a JSR \$A180. The routine at \$A180 jumps to the BRUN processor. So when your program finishes it will return to \$A17D, right after the JSR \$A180. From there it goes to \$9F83.

At \$9F83, DOS will finally exit from doing the BRUN command. If MON C is on, the carriage return from the end of the BRUN command will be echoed at this time. This can put you into a loop, however, because the BRUN command re-installed the DOS hooks in the input and output vectors. When the DOS hooks are installed, any character input or output will enter DOS first. Since we are still, in effect, inside DOS, because of the BRUN, we get into a loop. DOS is not re-entrant, as John Butterill put it. The BRUN command processor does a JSR \$A851, which re-installs the DOS hooks. If your program tries to do any character I/O through calls to \$FDED (COUT) or \$FD0C (RDKEY), and you start up your program by BRUNning it from inside an Applesoft program, you will get DOS into a loop. Or, even if your program does not do any I/O, if MONC is on DOS can still get into a loop.

I still think the easiest way to avoid this problem is to avoid using BRUN inside Applesoft programs. Use BLOAD and CALL instead. But sometimes you may want to use BRUN, because you do not know in advance where the CALL address would be. One way to allow I/O inside your own program even though it is to be BRUN from inside an Applesoft program is to disconnect or bypass the hooks. You could output characters by JSR \$FDF0, for example. But that would always go to the screen, and you may have a printer or an 80-column card or a modem hooked in, so that isn't a real solution. Another way is to dis-install the DOS hooks, by doing a JSR \$9EE0 or the equivalent. The code at \$9EE0 does this:

```
LDX #3
.1 LDA $AA53,X
  STA $36,X
  DEX
  BPL .1
  RTS
```

This unhooks DOS, but leaves any other I/O devices you have connected hooked in. After doing this step, your program can freely call COUT or RDKEY without DOS even knowing about it. You might also want to store a zero at \$AA5E, to turn off MONC. Your program can terminate then by a JMP \$3EA, which will restore the DOS hooks.



NEW !!! II IN A MAC: \$69.00

This Apple II emulator runs DOS 3.3 and PRODOS programs (including 6502 machine language routines) on a 512K Macintosh. All Apple II features are supported such as HI-RES/LO-RES graphics, 40/80 column text screens, language card and joystick. Also included: clock, RAM disk, keyboard buffer, on-screen HELP, access to the desk accessories and support for 4 logical disk drives. Package includes 2 MAC diskettes (PROGRAM holds emulation, communications and utility software, DATA holds DOS 3.3 and PRODOS system masters, including Applesoft and Integer BASIC) and 1 Apple II diskette (transfer software moves disk images to the MAC).

NEW !!! SCREEN.GEN: \$35.00

Develop HI-RES screens for your Apple II on a Macintosh. Don't be limited by MousePaint or other screen editors. Use MACPAINT (or any other application) on the MAC to create your Apple II screen. Then use SCREEN.GEN to transfer directly from the MAC to the Apple II (with SuperSerial card or equivalent). Package includes Apple II diskette with transfer software plus fully commented SOURCE code.

NEW !!! MIDI-MAGIC for Apple //c: \$49.00

Compatible with any MIDI equipped music keyboard, synthesizer, organ or piano. Package includes a MIDI-out cable (plugs directly into modem port - no modifications required!) and 6-song demo diskette. Large selection of digitized QRS player-piano music available for 19.00 per diskette (write for catalog). MIDI-MAGIC compatible with Apple II family using Passport MIDI card (or our own input/output card w/drum sync for only \$99.00).

FONT DOWNLOADER & EDITOR: \$39.00

Turn your printer into a custom typesetter. Downloaded characters remain active while printer is powered. Use with any Word Processor program capable of sending ESC and control codes to printer. Switch back and forth easily between standard and custom fonts. Special printer functions (like expanded, compressed etc.) supported. HIRES screen editor lets you create your own characters and special graphics symbols. For Apple II, II+, //e. Specify printer: Apple Dot Matrix, C.Itoh 8510A (Prowriter), Epson FX 80/100, or OkiData 92/93.

* The Font Downloader & Editor for the Apple Imagewriter Printer. For use with Apple II, II+, //e (with SuperSerial card) and the Apple //c (with builtin serial interface).

* FONT LIBRARY DISKETTE #1: \$19.00 contains lots of user-contributed fonts for all printers supported by the Font Downloader & Editor. Specify printer with order.

DISASM 2.2e : \$30.00 (\$50.00 with SOURCE Code)

Use this intelligent disassembler to investigate the inner workings of Apple II machine language programs. DISASM converts machine code into meaningful, symbolic source compatible with S-C, LISA, ToolKit and other assemblers. Handles data tables, displaced object code & even provides label substitution. Address-based triple cross reference generator included. DISASM is an invaluable machine language learning aid to both novice & expert alike. Don Lancaster says DISASM is "absolutely essential" in his ASSEMBLY COOKBOOK.

The 'PERFORMER' CARD: \$39.00 (\$59.00 with SOURCE Code)

Converts a 'dumb' parallel printer I/F card into a 'smart' one. Command menu eliminates need to remember complicated ESC codes. Features include perforation skip, auto page numbering with date & title. Includes large HIRES graphics & text screen dumps. Specify printer: MX-80 with Graftrax-80, MX-100, MX-80/100 with Graftraxplus, NEC 8092A, C.Itoh 8510 (Prowriter), OkiData 82A/83A with Okigraph & OkiData 92/93.

'MIRROR' ROM: \$25.00 (\$45.00 with SOURCE Code)

Communications ROM plugs directly into Novation's Apple-Cat Modem card. Basic modes: Dumb Terminal, Remote Console & Programmable Modem. Features include: selectable pulse or tone dialing, true dialtone detection, audible ring detect, ring-back, printer buffer, 80 col card & shift key mod support.

RAM/ROM DEVELOPMENT BOARD: \$30.00

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip (6116 type RAM for program development or 2716 EPROM to keep your favorite routines on-line). Maps into \$Cn00-CnFF and \$C800-CFFF.

C-PRINT For The APPLE //c: \$69.00

Connect standard parallel printers to an Apple //c. C-PRINT plugs into the standard Apple //c printer serial port and into any printer having a standard 36 pin centronics-type parallel connector. Just plug in and print!

Unless otherwise specified, all Apple II diskettes are standard (not copy protected!) 3.3 DOS.

Avoid a \$3.00 handling charge by enclosing full payment with order. VISA/MC and COD phone orders OK.

RAK-WARE 41 Ralph Road W. Orange N J 07052 (201) 325-1885



An alternative that seems to work is to save and restore the location where DOS saves the entering stack pointer. This is the culprit which causes the crippling loop. At \$9FB6, just before returning to whoever entered DOS, the stack pointer gets reset to the value it had when DOS was entered. If you enter DOS while you are still in DOS, the first value is replaced with the second. Then the final return point is lost, and it is loop-city. Your program can save and restore \$AA59, where the stack pointer is kept:

```
YOUR.PROGRAM
    LDA $AA59      save DOS stack pointer
    PHA
    LDA #0         turn off MON C
    STA $AA5E
```

...do all your stuff, including I/O

```
    PLA
    STA $AA59
    RTS
```

This method has the advantage that your program can issue its own DOS commands by printing them, the way you would from Applesoft. For example, the following program will work when BRUN from inside Applesoft.

```
        .OR $1000
        .TF B.SHOW OFF
DEMONSTRATE
    LDA $AA59
    PHA
    LDY #0          issue DOS CATALOG command
.l      LDA MSG,Y
    JSR $FDED
    INY
    CPY #MSGSZ
    BCC .l
    LDA #0
    STA $AA5E      "NOMON C"
    PLA
    STA $AA59
    RTS
MSG      .HS 8D.84
        .AS -/CATALOG/
        .HS 8D
MSGSZ    .EQ *-MSG

100 PRINT CHR$(4)"MONC"
110 PRINT CHR$(4)"BRUN B.SHOW OFF"
120 PRINT "FINISHED"
```

However, that program will not work correctly if you just type "BRUN B.SHOW OFF" from the command mode. You will get a syntax error after the catalog displays, because the catalog command is left in the input buffer incorrectly. Oh well!

Toggling Between Two Values.....Jan Eugenides

In the course of my job as Technical Editor for MicroSPARC, Inc. (the publishers of Nibble and Nibble Mac magazines), I am often called upon to modify programs that we are going to publish to make them compatible with configurations other than the one the author originally wrote for. Recently, I had to change a program to toggle between Drive 1 and Drive 3, rather than Drive 1 and Drive 2 as it was originally coded. Here is the original subroutine which toggled the drive number stored in a variable named CD:

```
TOGGLE.DRIVE
        LDA CD
        CMP #1
        BEQ .1
        LDA #1
        STA CD
        BNE .2
.1      INC CD
.2      RTS
CD      .BS 1
```

This code takes a total of 19 bytes, including the variable CD. My task was to exactly replace this routine with one which would toggle between 1 and 3 rather than 1 and 2. It had to use the same number of bytes, or less. It looks easy enough, but I couldn't come up with a solution. All my routines required one or two more bytes. I finally took the easy way out and patched it with a JMP to a free space near the end of the program, and put my code there. It works, but is there a shorter way?

Bob, you are the best code squeezer around, so I thought I'd give the problem to you. You'll undoubtedly come up with some sneaky code that does the trick in three bytes or less!

An Answer for Jan.....Bob Sander-Cederlof

I don't know if I am the best code squeezer or not, but I can't squeeze it all the way to three bytes! My best attempt is nine bytes:

```
TOGGLE.DRIVE
        LDA #1
CD      .EQ *-1
        EOR #2
        STA CD
        RTS
```

In general, you can toggle back and forth between any two values by using the EOR instruction. The toggle constant is simply the exclusive-or of the two values. For example, to toggle back and forth between the values \$A0 and \$B2, I would use "EOR #\$12".

My subroutine changes 1 to 3 and 3 to 1, as you requested.

However, it is not functionally identical to the original code. The original code did not store the variable CD inside an immediate-mode LDA, as I did. If that troubles you, simply change that line to "LDA CD" and add the line "CD .BS 1" at the end. The result takes ten bytes, still well under the limit.

The original code also always had the side-effect of setting carry status, so you might need to add a "SEC" instruction. I doubt it, because the original code would be very weird if it depended on this side-effect.

The original code not only changed 3 to 1, but also changed any other value not already 1 into 1. This is also probably not a necessary feature, because prior code should have made sure that we started with a valid drive number.

I came up with several other approaches to the problem. all of which are shorter than the original subroutine:

```
TOGGLE.DRIVE
        LSR CD      3 TO 1, OR 1 TO 0
        BNE .1      IT WAS 3 TO 1
        LDA #3      CHANGE 1 TO 3
        STA CD
.1      RTS
```

```
TOGGLE.DRIVE
        CLC
        LDA CD
        ADC #2      1 TO 3, OR 3 TO 5
        AND #3      5 TO 1
        RTS
```

None of these are particularly tricky or sneaky. In fact, the first and shortest one is the most straightforward. What would be tricky or sneaky is if the original author depended on the hidden side-effects in his subroutine.

Continued from page 5

		1640	PRINT.MSG		
087B-	68	1650	PLA	GET RETURN ADDRESS	
087C-	8D 86 08	1660	STA .1+1	LO-BYTE	
087F-	68	1670	PLA		
0880-	8D 87 08	1680	STA .1+2	HI-BYTE	
0883-	A0 01	1690	LDY #1		
0885-	B9 99 99	1700	LDA \$9999,Y	ADDRESS FILLED IN	
0888-	F0 06	1710	BEQ .2	...TERMINATING \$00	
088A-	20 ED FD	1720	JSR \$FDED	PRINT THE CHAR	
088D-	C8	1730	INY		
088E-	D0 F5	1740	BNE .1	...ALWAYS	
0890-	98	1750	TYA	ADJUST THE RETURN ADDRESS	
0891-	18	1760	CLC	BY ADDING THE MESSAGE LENGTH	
0892-	6D 86 08	1770	ADC .1+1		
0895-	A8	1780	TAY	SAVE LO BYTE FOR A WHILE	
0896-	A9 00	1790	LDA #0	THE HIGH BYTE TOO	
0898-	6D 87 08	1800	ADC .1+2		
089B-	48	1810	PHA		
089C-	98	1820	TYA		
089D-	48	1830	PHA		
089E-	60	1840	RTS	RETURN TO CALLER	
		1850	#-----		

The "Protocol Converter" is a firmware-controlled method of turning the //c disk port into a multi-drop peripheral bus able to support up to 127 external I/O devices. The bus connects devices which have enough intelligence: an "Integrated WOZ Machine" (IWM) chip, a 6502-type chip, RAM, and ROM. Data is transferred in a serial bit-stream at roughly 250,000 bits per second. So far, the only device anyone is building to run on the P/C bus is the UniDisk 3.5 from Apple.

As far as I have been able to determine, Apple's only published information about the protocol converter is in the Apple //c Technical Reference Manual, pages 114-142. The listing of the //c firmware in the same Manual also is informative. A preliminary document was available to developers, but most of the material is now given in the //c manual. Tom Weishaar ("Uncle DOS") promises a future article on the P/C in his "Open Apple" newsletter. (By the way, the June issue of "Open Apple" used the term "Smartport" as synonymous with "Protocol Converter".)

The Apple //e interface card for the UniDisk 3.5 also supports a "real" Protocol Converter. The Apple Memory Expansion Card, CirTech Flipster, and Applied Engineering RamFactor provide the same software interface with most of the features of the protocol converter for one I/O device (the memory card itself).

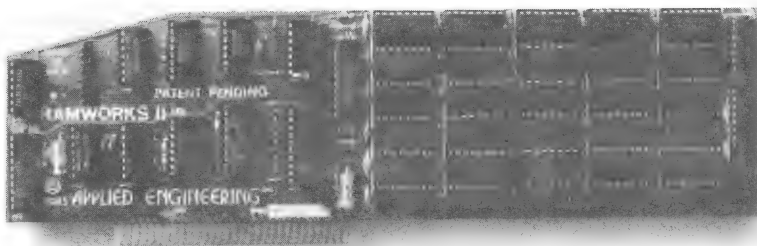
Apple briefly mentions the Protocol Converter in the Apple Memory Expansion Card manual (Appendix B, last paragraph), but warns against using it. They say "using the assembly-language protocol is fairly complicated". Nevertheless, a significant amount of the Apple firmware is used to implement the protocol converter features. It appears that someone inside Apple intends that the P/C will be included in the firmware of most future block-oriented devices. From a software stand-point, it could be used regardless of whether the actual hardware used the IWM-based bus, a SCSI bus, or no bus at all.

In order to use the protocol converter firmware, you need first to find it. The first step in finding it is to find which slot it is in. All of the cards with P/C firmware (so far) are also cards which control or emulate disk drives and have firmware supporting the ProDOS device driver protocol. Cards with ProDOS device driver firmware can be identified by four bytes: \$Cs01 = \$20, \$Cs03 = \$00, \$Cs05 = \$03, and \$Cs07 = \$00. The first three bytes in that list are the same for all disk drive controllers. The zero value at \$Cs07 distinguishes it as a disk controller with protocol converter firmware.

The next step is to find the entry point in the firmware for protocol converter calls. The byte at \$CsFF is the key. That byte is the offset in the firmware page for ProDOS calls. If \$CsFF = \$45, for example, ProDOS device driver calls would be "JSR \$Cs45". To get the address of the protocol converter entry point, add 3 to the ProDOS entry point. In my example, "JSR \$Cs48" would enter the protocol converter firmware. The actual value will probably be different for each kind of card, so you have to use software to find out what it is.

RamWorks II®

The Best Selling, Most Compatible, Most Recommended, Most Expandable Card Available.



64K to 16 MEG! RamWorks II Is Number One.

It's simple, RamWorks II sells the most because it does the most.

The AppleWorks Amplifier.

While RamWorks II is recognized by all memory intensive programs, NO other expansion card comes close to offering the multitude of enhancements to AppleWorks that RamWorks II does. Naturally, you'd expect RamWorks II to expand the available desktop, after all Applied Engineering was a year ahead of everyone else *including Apple* in offering more than 55K in AppleWorks and we still provide the largest AppleWorks desktops available. But a larger desktop is just part of the story. Just look at all the AppleWorks enhancements that even Apple's own card does not provide and *only* RamWorks II does. With a 256K or larger RamWorks II, all of AppleWorks will automatically load itself into RAM dramatically increasing speed by eliminating all the time required to access the program disk drive. Now switch from word processing to spreadsheet to database at the speed of light with no wear on disk drives.

Only RamWorks II eliminates AppleWorks' internal memory limits, increasing the maximum number of records available from 1,350 to over 15,000. *Only* RamWorks II increases the number of lines permitted in the word processing mode from 2,250 to over 15,000. And *only* RamWorks II (256K or larger) offers a built-in printer buffer, so you won't have to wait for your printer to stop before returning to AppleWorks. Ram-

Works II even expands the clipboard. And auto segments large files so they can be saved on two or more disks.

RamWorks II, nothing comes close to enhancing AppleWorks so much.

The Most Friendly, Most Compatible Card Available.

Using RamWorks II couldn't be easier because it's compatible with more off-the-shelf software than any other RAM card. Popular programs like AppleWorks, Pinpoint, Catalyst, MouseDesk, Howard-Soft, FlashCalc, The Spread Sheet, Managing Your Money, SuperCalc 3a, and MagiCalc to name a few (and *all* hardware add on's like ProFile and Sider hard disks). RamWorks II is even compatible with software written for Apple cards. But unlike other cards, RamWorks II plugs into the IIe auxiliary slot providing our super sharp 80 column text in a completely integrated system while leaving expansion slots 1 through 7 available for other peripheral cards.

Highest Memory Expansion.

Applied Engineering has always offered the largest memory for the IIe and RamWorks II continues that tradition by expanding to 1 full MEG on the main card using standard RAMs, more than most will ever need (1 meg is about 500 pages of text)...but if you do ever need more, RamWorks II has the widest selection of expander cards available. Additional 512K, 2 MEG, or multiple 16 MEG cards just snap directly onto RamWorks II by plugging into the

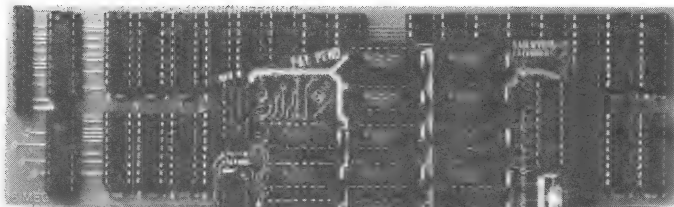
industry's only low profile (no slot 1 interference) fully decoded memory expansion connector. You can also choose non-volatile, power independent expanders allowing permanent storage for over 20 years.

It Even Corrects Mistakes.

If you've got some other RAM card that's not being recognized by your programs, and you want RamWorks II, you're in luck. Because all you have to do is plug the memory chips from your current card into the expansion sockets on RamWorks II to recapture most of your investment!

The Ultimate in RGB Color.

RGB color is an option on RamWorks II and with good reason. Some others combine RGB output with their memory cards, but that's unfair for those who don't need RGB *and* for those that do. Because if you don't need RGB Applied Engineering doesn't make you buy it, but if you want RGB output you're in for a nice surprise because the RamWorks II RGB option offers better color graphics plus a more readable 80 column text (that blows away any composite color monitor). For only \$129 it can be added to RamWorks II, giving you a razor sharp, vivid brilliance that most claim is the best they have ever seen. You'll also appreciate the multiple text colors (others only have green) that come standard. But the RamWorks II RGB option is more than just the ultimate in color output because unlike others, it's fully



16 MEG Expander

512K Expander

compatible with all the Apple standards for RGB output control, making it more compatible with off-the-shelf software. With its FCC certified design, you can use almost any RGB monitor because only the new RamWorks II RGB option provides both Apple standard and IBM standard RGB outputs (cables included). The RGB option plugs into the back of RamWorks II with no slot 1 interference (works on the original RamWorks, too) and remember you can order the RGB option with your RamWorks II or add it on at a later date.

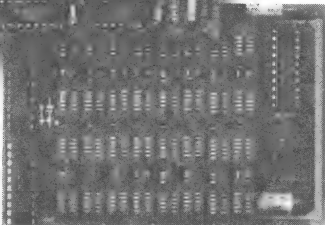
True 65C816 16 Bit Power.

RamWorks II has a built-in 65C816 CPU port for direct connection to our optional 65C816 card. The only one capable of linearly addressing more than 1 meg of memory for power applications like running the Lotus 1-2-3™ compatible program, VIP Professional. Our 65C816 card does not use another slot but replaces the 65C02 yet maintains full 8 bit compatibility.

Endorsed by the Experts.

Steve Wozniak, creator of the Apple Computer said "I wanted a memory card for my Apple that was fast, easy to use, and very compatible, so I bought RamWorks." A+ magazine said "Applied Engineering's RamWorks is a boon to those who must use large files with AppleWorks...I like the product so much that I am buying one for my own system." inCider magazine said "RamWorks II is the most powerful auxiliary slot memory card available for your Iie, and I rate it four stars...For my money, Applied Engineering's RamWorks II is king of the hill"

Apple experts everywhere are impressed by RamWorks II's expandability, versatility, ease of use, and the sheer power and speed that it adds to any Iie. With a RamWorks II in your Apple, you'll make IBM PC's and AT's look like slowpokes.

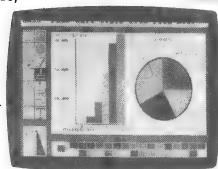
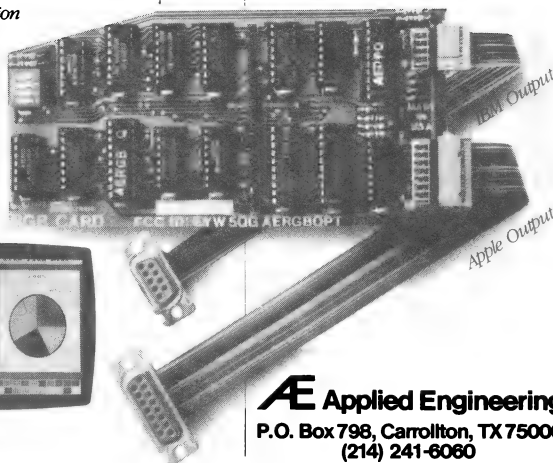


2 Meg Expander

It's Got It All

- 15 day money back guarantee
- 5 year hassle free warranty insures coverage no matter where you purchase
- Built-in super sharp 80 column display, (with or without RGB)
- Expandable to 1 MEG on main card
- Expandable to 16 meg with expander card, with NO slot 1 interference
- Can use 64K or 256K RAMs
- Powerful linear addressing 16 bit coprocessor port
- Automatic AppleWorks expansion up to 3017K desktop
- Accelerates AppleWorks
- Built-in AppleWorks printer buffer
- The only large RAM card that's 100% compatible with all Iie software

RGB Option



- RamDrive™ the ultimate disk emulation software included free
- Memory is easily partitioned allowing many programs to be in memory at once
- Compatible, RGB option featuring ultra high resolution color graphics and multiple text colors, with cables for both Apple and IBM type monitors
- Built-in self diagnostics software
- Lowest power consumption (patent pending)
- Takes only one slot (auxiliary) even when fully expanded
- Software industry standard
- Advanced Computer Aided Design
- Used by Apple Computer, Steve Wozniak and virtually all software companies
- Displays date and time on the AppleWorks screen with any PRO-DOS compatible clock
- Much, much more!

RamWorks II with 64K	\$179
RamWorks II with 256K	\$219
RamWorks II with 512K	\$269
RamWorks II with 1 MEG	\$369
RamWorks II with 1.5 MEG	\$539
RamWorks II with 3 to 16 MEG	CALL
65C816 16 Bit Card	\$159
RGB Option	\$129
256K Upgrade	\$ 50

RamWorks II. The industry standard for memory expansion of the Apple Iie.

ORDER YOUR RamWorks II TODAY.

9 a.m. to 11 p.m. 7 days, or send check or money order to Applied Engineering MasterCard, Visa and C.O.D. welcome. Texas residents add 5% sales tax. Add \$10.00 if outside USA.

AE Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

A program to find the slot and build the address of the protocol converter could look like this:

```

pcaddr .eq $01,$02
find.pc lda #0
      sta pcaddr
      ldx #C7      slot = 7 to 1 step -1
.1      stx pcaddr+1
      ldy #7
.2      lda (pcaddr),y  $Cs07,05,03,01
      cmp pc.sig,y
      beq .3
      dex
      cpx #C1
      bcs .1      try next slot
      sec          signal could not find pc
      rts

.3      dey
      dey
      bpl .2
      lda (pcaddr),y  $CsFF
      adc #2      carry was set
      sta pcaddr
      rts          carry clear signals pc found

pc.sig .HS FF.20.FF.00.FF.03.FF.00

```

Once you have the address of the protocol converter firmware, you call it in a manner similar to ProDOS MLI calls. You must plug the address of the protocol converter entry into a "JSR" instruction, which is followed by a one-byte command code and a two-byte address. The command code is a number from \$00 to \$09 which specifies which action you want the protocol converter to take. The address is the address of a parameter block, which provides additional information for processing the command, or a place for the information returned by the command. After the protocol converter has finished processing your command, it returns control to the next byte after the pointer to the parameter block. If carry is clear, there was no error. If carry is set, the A-register contains an error code.

Since my FIND.PC program left the address in two page zero locations, we could simply put a JMP opcode (\$4C) in front of the address to make it into a JMP instruction. Then our calls to the protocol converter would look like this:

```

callpc .eq $00      (just before pcaddr)
      jsr find.pc
      bcs ...      ...no pc found
      lda #$4C      JMP opcode
      sta callpc
      ...      ...other code
      jsr callpc
      .da #cmd,parameters
      ...      ...more code

```

Apple warns programmers NOT to use any page zero locations when calling the protocol converter firmware, saying that some page

To boldly go at speeds no Apple has gone before.



Get TransWarp™. The fastest accelerator you can buy for your Apple™ IIe, II, or II+.

Computing at warp speed!

It's an experience you shouldn't miss. And with TransWarp, you won't have to. Because TransWarp will run your software up to 3.6 times faster — leaving other accelerators in the stardust!

No more yawning while your Apple™ slowly rearranges text or calculates spreadsheets. With 256K of ultra-fast RAM, TransWarp speeds up *all* Apple software — including AppleWorks, SuperCalc, Xit Visicalc, and all educational software, graphics and games. And it's compatible with all standard peripheral cards (such as Ram-Works II and Apple memory cards), Profile, and Sider hard disks, 3 1/2" disk drives, 80 column cards, modems, clock cards, mouses and more! You name it, TransWarp accelerates it. There's even a 16 bit upgrade chip available should 16 bit software become available for the Apple.



*"I recommend Applied
Engineering products
wholeheartedly."*

*Steve Wozniak, the creator
of Apple Computer*

An important difference.

TransWarp's not the only speedup card on the market. But it's the only one that accelerates your Apple's main memory, ROM *and* auxiliary memory. And with more and more programs residing in auxiliary memory, buying anyone else's accelerator makes less and less sense. TransWarp even works with most DMA devices including the Swift™ card.

There's more difference. Since TransWarp doesn't use memory caching, it accelerates *all* software — and not just most of it.

A cinch to use.

Simply plug TransWarp into any slot in your Apple II, II+ or IIe including slot 3 in the IIe. Instantly you'll be computing at speeds you only dreamed about before. And should you ever wish

to run at normal speed, simply press the ESC key while turning your Apple on.

Since TransWarp is completely transparent, you won't need pre-boot disks or special software. It's ready to go right out of the package!

Speed = Productivity

Imagine the productivity gains you'll achieve when your programs are running over three times faster. TransWarp is so powerful, your Apple will make IBM PCs™ and even ATs™ look like slowpokes — whether you're planning taxes, plotting charts or playing games! Take a look at a few of the features that set TransWarp apart:

- 3.6 MHz 65C02
- 256K of ultra-fast on-board RAM
- Accelerates main *and* auxiliary memory
- Low power consumption for cool operation
- Totally transparent operation with all software
- Plugs into any slot, including slot 3 on the Apple IIe
- Accelerated 16 bit option available

Satisfaction guaranteed!

Give your Apple the TransWarp advantage. With our risk-free 15-day money back guarantee, you have nothing to lose but wasted time. Call today!

TransWarp Accelerator
16 bit upgrade (may add later)

\$279
\$89

For fast response:

Call Applied Engineering, 9 a.m. to 11 p.m., 7 days at (214) 241-6060. MasterCard, VISA and C.O.D. welcome. Texas residents add 5% sales tax. Add \$10.00 if outside U.S.A.

Or mail check or money order to Applied Engineering, P.O. Box 798, Carrollton, TX 75006.

AE Applied Engineering
The Apple enhancement experts.

P.O. Box 798, Carrollton, TX 75006 (214) 241-6060

zero locations are used by that firmware. They do not say what locations they use, but my investigations show that they use bytes in the range from \$40 to \$4F. What they do is push those on the stack, put in their own data, and at the end restore the original contents from the stack. They use an awful lot of stack, up to 35 bytes. (The RamFactor firmware uses no more than 17 bytes of stack for protocol converter calls, including the two used by your JSR.) If you want be safe rather than possibly sorry, you can copy the PCADDR bytes up into your own program. You could even plug them into every JSR which calls protocol converter. A cleaner way might be like this:

```

        jsr find.pc
        bcs ...           ...no pc found
        lda pcaddr
        sta callp+1
        lda pcaddr+1
        sta callpc+2
        ...
        jsr callpc
        .da #cmd,parameters
        ...
callpc  jmp *   address filled in

```

Description of Protocol Converter Commands

Apple defines ten commands for the protocol converter firmware. These are not necessarily identical in function for all devices which use the protocol converter. In fact, Apple's memory card uses two of the commands differently than the UniDisk 3.5 does. The protocol converter firmware in the RamFactor functions exactly the same as that in the Apple Memory Expansion Card.

The following chart summarizes the ten commands as implemented in the Apple Memory Expansion Card and RamFactor firmware. A more detailed description of each command follows the chart. I am particularly pointing this at the memory cards rather than the Unidisk 3.5, because I believe these cards will be more popular with hackers like you and me. Furthermore, the Unidisk 3.5 information is available in the //c manual, but Apple has not released this detail for owners of the memory card.

	Parameters:	+0	+1	+2	+3	+4	+5	+6	+7	+8
	cmd	cnt	unit							
PC Status	\$00	3	0	buf1	bufh	code				
RAM Status	\$00	3	1	buf1	bufh	code				
Read Block	\$01	3	1	buf1	bufh	blkh	blk1	blk2	blk3	
Write Block	\$02	3	1	buf1	bufh	blkh	blk1	blk2	blk3	
Format	\$03	1	1							
Control	\$04	3	0/1	buf1	bufh	code				
Init	\$05	1	0/1							
Read Bytes	\$08	4	1	buf1	bufh	cnth	cnt1	adrh	adrm	adrl
Write Bytes	\$09	4	1	buf1	bufh	cnth	cnt1	adrh	adrm	adrl

Error Codes \$01 Command not \$00-\$05,\$08, or \$09
 \$04 Wrong parameter count
 \$11 Invalid Unit Number
 \$21 Invalid Status or Control code
 \$2D Block Number too large

PC Status (cmd \$00, unit \$00, code \$00): reads the status of the protocol converter itself into your buffer. The status of a memory card is always 8 bytes, with the first byte = \$01 and all the others = \$00. Also returns with \$08 in the X-register and \$00 in the Y-register. (\$0008 is the number of bytes stored in your buffer.) This is of value only for compatibility with other devices supporting protocol converter firmware.

RAM Status (cmd \$00, unit \$01, code \$00 or \$03): reads the status of the memory card into your buffer. Code \$00 stores four bytes: the first is always \$F8, and the other three are the number of blocks in the current partition (lo, mid, hi order). (Y,X) will equal (\$00,\$04) when it is finished, showing that four bytes were stored. Code \$03 will store 25 bytes: the first four are the same as code \$00 returned; the next 17 are the name of the card in "ProDOS Volume Name" format (length of name in first byte, ASCII characters of name with hi-bit off, padded with blanks); and finally, four zero bytes. The card name is "RAMCARD". (Y,X) will return (\$00,\$19) when finished, indicating that 25 bytes were stored.

Obviously, the Status commands will operate differently on a real P/C bus, and the actual details will vary according to the device you interrogate.

Read Block (cmd \$01): reads the specified block from the memory card. (In RamFactor, the block number is relative, inside the currently selected RamFactor partition.) You can read a block into a buffer in //e Auxiliary Memory by calling the P/C with the RAMWRT soft-switch set to AuxMem.

Write Block (cmd \$02): writes the specified block from your buffer into the memory card. (In RamFactor, the block number is relative, inside the current RamFactor partition.) If you are careful and follow all the rules, you can write a block from a buffer in Auxiliary Memory by calling the protocol converter with the RAMRD soft-switch set to AuxMem. You have to put the code that sets the RAMRD switch and calls the protocol converter, and its parameter block, in zero-page or stack-page motherboard RAM (\$0000-01FF), or in the language card RAM area. Or, you can have both RAMRD and RAMWRT set for AuxMem and be executing a program from within AuxMem. I always have a conceptual battle dealing with this kind of bank switching.

Format (cmd \$03): does nothing in a memory card.

Control (cmd \$04): does nothing in a memory card. If the code is not \$00, you get error code \$21. The buffer is never used.

Init (cmd \$05): does nothing in a memory card.

Open or Close (cmd \$06 or \$07): cause error code \$01 in a memory card. These commands only apply to character-oriented devices, and memory is a block-oriented device (so says Apple). Maybe someday someone will build a peripheral which is character-oriented and includes P/C firmware.

Read Bytes (cmd \$08): reads a specified number of bytes starting at a specified memory-card address into your buffer.

	VIDEO SAPPHIRE	SMART MST UPGRADE	LOW COST DESIGN	NO CABLES NEEDED	NO 100' MATERIAL	SMART MST FOR TV'S	NO CABLES OR WIRING	NO WIRE OR CABLE
VIEWMASTER 80	x					x	x	
SI PRTERM		x						
WIZARD 80							x	
VISION 80	x	x						
OMNIVISION		x					x	
VIEWMAX 80	x	x						
SMARTERM	x	x						
VIDEOTERM								

One look at the chart will give you some of the reasons there's only one smart choice in 80 column cards for your Apple. But the real secret to Viewmaster 80's success is something even better: Total compatibility.

And the Viewmaster 80 delivers a super sharp, state-of-the-art display with a 7x9 character matrix for clear, easily readable characters. Here are just a few of the powerful features the Viewmaster 80 delivers for a great price (\$139):

- Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5½% sales tax. Add \$10.00 outside U.S.A.

Page 22.....Apple Assembly Line.....June, 1986.....Copyright (C) S-C SOFTWARE

The byte count may be as high as \$FFFF, but this would obviously wreak havoc inside your Apple. No checks are made inside the protocol firmware for reasonableness of the buffer address or the byte count, so be careful. You would NEVER read into a buffer in the I/O address range (\$C000-\$CFFF).

The memory-card address may be as high as \$7FFFFFFF. (In RamFactor, the address is relative inside the current partition.) This corresponds to a total of 8 megabytes, which is only half the maximum capacity of a RamFactor card. Apple has arbitrarily limited us to this maximum, because they use the top bit of the card address to specify whether the buffer is in MainMem (bit 23 = 0) or AuxMem (bit 23 = 1). (Bit 23 of the address is bit 7 of the last byte of the parameter block.)

Write Bytes (cmd \$09): writes a specified number of bytes from your buffer starting at a specified memory-card address. The details of byte count, buffer location, and memory-card address are the same as for the Read Bytes (\$08) command.

The Unidisk 3.5 firmware interprets commands \$08 and \$09 differently. Unidisk uses this pair to read and write Macintosh disks, which have 524-byte blocks.

All of the RamFactor protocol converter commands operate within the current active partition. In the Apple card there is only one partition (the whole card). RamFactor has nine partitions, and you are always in one of them. If you start with a blank card, the first call to the RamFactor protocol converter will set up the first partition with all but 1024 bytes, make that partition the current active one, and empty all the others.

Bill Morgan's articles on interfacing the Unidisk 3.5 with DOS 3.3 illustrate the use of protocol converter calls with that device. The real power of the protocol converter concept will not be realized until a variety of devices are available which use it. Maybe its real future is bound up in the new 65816-based Apple //.

RAMWORKS™

**ACCEPT NO SUBSTITUTES.
BECAUSE THERE AREN'T ANY.**

There's only one card like RamWorks. We've got the best hardware design. We supply the best software and we've got the best support from software companies.

If someone tempts you with an imitation, please get both sides of the story. You'll discover why RamWorks offers the best enhancements to AppleWorks and other programs, and at the lowest price.

GUARANTEED!

214-241-6060
9 AM - 11 PM

AE
"We Set the Standard"
APPLIED ENGINEERING

The ProDOS Machine Language Interface (MLI) returns an error code in the A-register if anything goes wrong. There are about 30 error codes, with values from \$01 to \$5A. BASIC.SYSTEM reduces the number of different error codes to 18, calling many of them simply "I/O ERROR". A nearly complete description of the error codes can be found in several references:

- "Apple ProDOS--Advanced Features", pages 68-70.
- "Beneath Apple ProDOS", pages 6-59 thru 6-61.
- "ProDOS Technical Reference Manual", pages 77-79.

When I am working with a new program which has a lot of MLI calls, it is helpful to have one central error handler to print out the error information. Gary Little gives us such a subroutine on pages 66 and 67 of his "Apple ProDOS -- Advanced Features." Gary's program prints the message "MLI ERROR \$xx OCCURRED AT LOCATION \$yyyy", where xx is the hexadecimal error code and yyyy is the address of the next byte after the MLI call. You can mentally subtract 6 from the yyyy address to get the actual address of the JSR \$BF00 that caused the error.

I assume you already know, if you are following me this far, that MLI calls take the form "JSR \$BF00", followed by three data bytes. The first data byte is the opcode, and the other two are the address of the parameter block for the MLI call:

```
JSR $BF00
.DA #OPCODE,PARAMETERS
```

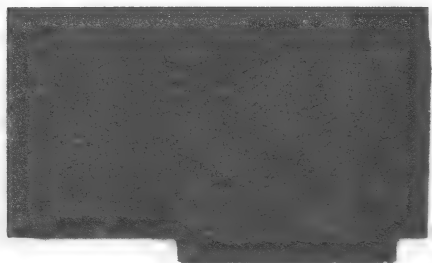
It would be nice if the general error handler would give us a little more information. First, I would like for it to print out the actual address of the JSR \$BF00, rather than the return address. Second, I would like for it to print out the three bytes which follow the JSR \$BF00.

First, I recoded Gary's routine so that it took a lot less space. (Littler than Little's!) I shortened the message and tightened the code. My version prints simply "AT" in place of "OCCURRED AT LOCATION." Then I used a message printing subroutine to print the two text strings, rather than the two separate loops he used. His took 83 bytes, mine only 56.

```

-----
1000 *SAVE MLI.ERROR
1010 *-----
BF9C- 1020 CMDADR .EQ $BF9C
1030 *-----
F941- 1040 PRNTAX .EQ $F941
FD8E- 1050 CROUT .EQ $FD8E
FDDA- 1060 PRBYTE .EQ $FDDA
FDED- 1070 COUT .EQ $FDED
1080 *-----
1090 MLI.ERROR
0800- 48 1100 PHA
0801- A0 00 1110 LDY #QERR SAVE ERROR CODE
0803- 20 1F 08 1120 JSR PRMSG
0806- 68 1130 PLA
0807- 20 DA FD 1140 JSR PRBYTE
080A- A0 0D 1150 LDY #QAT
080C- 20 1F 08 1160 JSR PRMSG
080F- AD 9D BF 1170 LDA CMDADR+1
0812- AE 9C BF 1180 LDX CMDADR
0815- 20 41 F9 1190 JSR PRNTAX
0818- 4C 8E FD 1200 JMP CROUT
```


With Z-80 Plus,TM run CP/M—the largest body of software in existence.



*Now, get two computers in one,
and all the advantages of both.*

Enter the CP/M world with the new Z-80 Plus card from Applied Engineering, and introduce your Apple IIe[®] or II + [®] to the thousands of CP/M programs. Only the Z-80 Plus comes standard with the new 4.0 software, the most advanced system ever for running CP/M programs.

The new 4.0 boasts advanced features like built-in disk emulation for popular memory expansion boards, boosting both system speed and storage capacity. And menu-driven utilities that let you get to work faster. The Z-80 Plus also lets you run older CP/M programs — all the way down to Version 1.6 (2.2 is the most popular).

The Z-80 Plus is the only card on the market capable of accessing more than 64K in an Apple IIe. If you have an extended 90-column card, all 128K is usable, and if you have RamWorks, up to 1088K is available.

Each Z-80 Plus comes with our CP/M Ram Drive software, enabling IIe owners to use an extended 80-column card or a RamWorks card as a high-speed Ram disk which runs CP/M software up to *twenty times faster*. So packages like WordStar and dBASE II run at blinding speed.

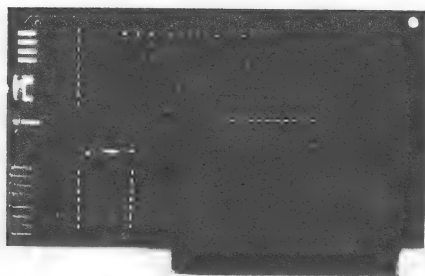
Simply plug the Z-80 Plus into any slot in your Apple. You'll get the benefits of two computers in one — all at an unbelievably low price (only \$139).

- Fully compatible with ALL CP/M software
- Fully compatible with most hard disks, including Corvus and the Sider
- Fully compatible with Microsoft disks (no pre-boot required)
- Specifically designed for high speed operation in the Apple IIe (runs just as fast in the Apple II + and Franklin)
- Runs WordStar, dBASE II, Turbo Pascal, Fortran-80, Peachtree and ALL other CP/M software with *no pre-boot*
- Semi-custom I.C. and low parts count allows Z-80 Plus to fly through CP/M programs with extremely low power consumption (we use the Z-80B)
- Does EVERYTHING other Z-80 boards do, *plus* Z-80 interrupts
- Five year warranty

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering, MasterCard, VISA and C.O.D. welcome. Texas residents add 5½% sales tax. Add \$10.00 outside U.S.A.

AE Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

Timemaster H.O.,TM the only clock that displays time and date on AppleWorksTM screens and files.



*Now, get all the features of
all the competition combined!*

It's the smart way to put the time and date on your Apple II + [®] or IIe[®]. Because only the Timemaster H.O. packs *ALL* the features of all the competition *combined*, including leap year, year (not just in PRO-DOS), month, date, day of week, hours, minutes, seconds and milliseconds. It's totally PRO-DOS, DOS 3.3, PASCAL and CP/M compatible. And of course, it works better than any other clock with AppleWorks.

If you're using or writing software for other clock cards, you're still covered. Because the H.O. will *automatically* emulate them. And the Timemaster H.O. adds 14 new commands to BASIC. The H.O. even comes complete with two disks full of sample programs, including a computerized appointment book, a DOS dating program, interrupt programs, and over 30 programs that others charge extra for — or don't even offer.

As a low-cost option, you can add true BSR remote control to the H.O., giving you remote control of up to 16 lights and appliances in your home or office.

- Fully PRO-DOS and DOS 3.3, CP/M and PASCAL compatible
- Time in hours, minutes, seconds and milliseconds (the ONLY PRO-DOS compatible card with millisecond capability); date with year, month, day of week and leap year
- 24-Hour military format or 12-hour AM/PM format
- Eight software controlled interrupts so you can run two programs at the same time (many examples included)
- Allows AppleWorks to time and date stamp all data automatically
- The only clock card that displays time and date on the AppleWorks screen
- Five year warranty

Clock price	\$129.00
BSR option (may be added later)	\$ 49.00

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering, MasterCard, VISA and C.O.D. welcome. Texas residents add 5½% sales tax. Add \$10.00 outside U.S.A.

AE Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

```

081B- 20 ED FD 1220 MSG1 JSR COUT
081E- C8 1230 INY
081F- B9 25 08 1240 PRMSG LDA MSGS,Y
0822- D0 F7 1250 BNE MSG1
0824- 60 1260 RTS
1270 *-----
1280 MSGS
1290 QERR .EQ *-MSGS
1300 .HS 8D
00-
0825- 8D
0826- CD CC C9
0829- A0 C5 D2
082C- D2 CF D2
082F- A0 A4 1310 .AS -/MLI ERROR $/
0831- 00 1320 .HS 00
0D- 1330 QAT .EQ *-MSGS
0832- A0 C1 D4
0835- A0 A4 1340 .AS -/ AT $/
0837- 00 1350 .HS 00
1360 *-----

```

Next, I started adding the features I mentioned above. The final program takes 92 bytes, which is 9 more than Gary's. It displays the error message "MLI ERROR \$xx AT \$yyyy (op.addr)."

Lines 1080-1160 pick up the address MLI saved in the System Global Page, and subtract six from it. The result is stored into the LDA \$9999,Y instruction at line 1200. Horrors! Self-modifying code! The loop at lines 1180-1240 copies the three data bytes which follow the JSR \$BF00 into the three variables at lines 1390-1410.

Lines 1260-1360 print out the error message. This loop differentiates between ASCII characters (bit 7 = 1) and data offsets (bit 7 = 0). The text to be printed is in lines 1430-1550. Note that I used the negative ASCII form for the text, and .DA lines for the data bytes which will be printed in hexadecimal. The expressions in those .DA lines compute an offset from the beginning of the subroutine, which will come out as a value less than \$7F. I also used the value 00 to terminate the entire message. The \$8D bytes are RETURN characters, to make sure the error message prints on a line by itself.

```

1000 *SAVE MLI.ERROR.PLUS
1010 *-----
BF9C- 1020 CMDADR .EQ $BF9C
1030 *-----
FDDA- 1040 PRBYTE .EQ $FDDA
FD4D- 1050 COUT .EQ $FD4D
1060 *-----
1070 MLI.ERROR.PLUS
0800- 8D 3B 08 1080 STA ERRCOD SAVE ERROR NUMBER
0803- AC 9D BF 1090 LDY CMDADR+1
0806- AD 9C BF 1100 LDA CMDADR SUBTRACT 6 FROM ADDRESS
0809- 38 1110 SEC
080A- E9 06 1120 SBC #6
080C- 8D 1A 08 1130 STA CALADR+1 CALL ADDR LO
080F- B0 01 1140 BCS .1
0811- 88 1150 DEY
0812- 8C 1B 08 1160 .1 STY CALADR+2 CALL ADDR HI
1170 *-----
0815- A0 02 1180 LDY #2
0817- A2 03 1190 LDX #3 COPY OPCODE & PARMS ADDR
0819- BD 99 99 1200 CALADR LDA $9999,X (ADDRESS FILLED IN)
081C- E8 1210 INX
081D- 99 3C 08 1220 STA PARMADR.H,Y
0820- 88 1230 DEY
0821- 10 F6 1240 BPL CALADR ...UNTIL Y=-1
1250 *-----

```

```

0823- 30 03      1260      BMI .2      ...ALWAYS
0825- 20 ED FD   1270 .1      JSR COUT
0828- C8          1280 .2      INY
0829- B9 3F 08   1290      LDA QERR,Y
082C- 30 F7      1300      BMI .1      ...ASCII CHAR
082E- D0 01      1310      BNE .3      ...DATA BYTE
0830- 60          1320      RTS      ...END
0831- AA          1330 .3      TAX      USE AS INDEX
0832- BD 00 08   1340      LDA MLI.ERROR.PLUS,X
0835- 20 DA FD   1350      JSR PRBYTE
0838- 4C 28 08   1360      JMP .2      NEXT CHAR
083B-          1370 *-----
083B-          1380 ERRCOD .BS 1
083C-          1390 PARMADR.H .BS 1
083D-          1400 PARMADR.L .BS 1
083E-          1410 OPCODE .BS 1
083F-          1420 *-----
0840- 8D          1430 QERR .HS 8D
0840- CD CC C9
0843- A0 C5 D2
0846- D2 CF D2
0849- A0 A4      1440      .AS -/MLI ERROR $/
084B- 3B          1450      .DA #ERRCOD-MLI.ERROR.PLUS
084C- A0 C1 D4
084F- A0 A4      1460      .AS -/ AT $/
0851- 1B          1470      .DA #CALADR-MLI.ERROR.PLUS+2
0852- 1A          1480      .DA #CALADR-MLI.ERROR.PLUS+1
0853- A0 A8      1490      .AS -/ (/
0855- 3E          1500      .DA #OPCODE-MLI.ERROR.PLUS
0856- AE          1510      .AS -/./
0857- 3C          1520      .DA #PARMADR.H-MLI.ERROR.PLUS
0858- 3D          1530      .DA #PARMADR.L-MLI.ERROR.PLUS
0859- A9          1540      .AS -/)/
085A- 8D 00      1550      .HS 8D.00
085A-          1560 *-----

```

NEW FROM DON LANCASTER

HANDS-ON BOOKS

Apple Assembly Cookbook	21.50
All About Appewriter	12.50
Appewriter Cookbook	19.50
Enhancing your Apple vol I	15.50
Enhancing your Apple vol II	15.50
Micro Cookbook vol I	15.50
Micro Cookbook vol II	15.50
CMOS Cookbook	14.50
TTL Cookbook	12.50
TV Typewriter Cookbook	12.50
Active Filter Cookbook	14.50
Incredible Secret Money Machine	7.50

UNLOCKED SOFTWARE

Absolute Reset Ite & Iic	19.50
Appewriter Toolkit (Dos 3.3e)	39.50
Appewriter Toolkit (ProDOS)	39.50
Both Appewriter Toolkits	59.50
Appewriter/Laserwriter Utilities	39.50
Laserwriter Demo Pack	FREE
Appleworks Disassembly Script	49.50
Enhance vol I Companion Disk	19.50
Enhance vol II Companion Disk	19.50
Assembly CB Companion Disk	19.50
Appewriter CB Companion Disk	19.50
Classic Cell Animation Demo	12.50

FREE VOICE HELPLINE

VISA/MC

SYNERGETICS

Box 809-SC
Thatcher, AZ 85552
(602) 428-4073

1200 BAUD MODEMS

Coit Valley Computers has two modems for your every need. Both are top quality state-of-the-art 1200/300/110 baud Hayes™ compatible modems, which means your computer can send & receive data at lightning fast speeds! And automatically switch between 1200 and 300 baud to communicate with slower Apples. Since neither comes with software, we carry Ascii Express PRODOS at a low price of \$89.

AVATEX™ 1200 EXTERNAL STAND-ALONE MODEM

\$199.

- 100% plug in Modem for **Apple IIc or Macintosh** with proper cable (see below)
- Universal modem that only requires modem compatible serial card (or port), & cable, to plug into **Apple IIe, Apple II+, or IBM**
- Auto Answer, Auto Dial, Auto Redial, Auto Disconnect
- Full Bell 212A compatibility
- Automatically switches between 300 baud & 1200 baud incoming speeds
- Complete diagnostics & full complement of LEDS (TR, SD, RD, HS, MC, TM, RI)
- DATA/VOICE Button switches from talk to data transmission & back again
- FREE CompuServe offer & free access time. One year warranty.

CERMETEK APPLE-MATE™ 1200 INTERNAL MODEM

\$209.

- Internal 1200 baud modem for **Apple IIe or Apple II+**
- Only one card & takes only one slot w/ no external interface or power supply

- Built-in Super Serial Card equivalent
- 1200/300/110 baud operation and Bell 212A compatibility
- Built-in Speaker & Diagnostics
- Auto Dial, Auto Answer, & Auto Select. Two year warranty.

2400 BAUD MODEMS — Call

CABLES REQUIRED WITH AVATEX MODEMS

Apple IIc - AvateX Cable	\$ 22.
Apple IIe, II+ - AvateX Cable	25.
Macintosh - AvateX Cable	27.
IBM - AvateX Cable	23.

OTHER APPLE PERIPHERALS

IIe/II+ Serial Modem Card	\$ 99.
RGB Monitor for Apple IIe	324.
RGB Monitor to Apple Cable	24.
Multiram RGB cards (facing page)	➡ ➡ ➡

With prices this low, how can you afford to be without a 1200 baud modem? Just the savings in connect time, will pay for the difference between a 300 & 1200 baud modem. You can get everything you need from Coit Valley Computers. Shipping on modems \$5-Ground/\$8-Air, monitors \$10. See terms on facing page.

Hayes, AvateX, Apple-Mate respective registered trademarks of Hayes Microcomputer Products, E + E DataComm, Cermetek Micro.

COIT VALLEY COMPUTERS • 14055 Waterfall Way, Dallas, TX 75240 • (214) 234-5047

6 Meg Iie/640k Iic

Don't buy yesterday's card that doesn't offer battery backed-up RAM or 65C816 new Apple technology just because it's advertised a lot! You can buy Checkmate Technology's **State-Of-The-Art MULTIRAM RGB RAM CARD™** with **BATTERY BACKED-UP STATIC RAM** options that can load & save programs (like AppleWorks, for 10 years!) It is a **FASTER & LESS EXPENSIVE REPLACEMENT FOR HARD DISKS**, is **USER EXPANDABLE TO 6 MEGABYTES**, compatible with all (100%) 3rd party software hardware, has an optional real 16-Bit 65C816 slot saver Co-Processor card, sharp 80 columns, super Double Hi-Res, & **BUILT IN RGB™**! It's a direct substitute for Ramworks II™ or Apple Ext 80 column cards & has an amazing 5 year warranty! Unlike Ramworks II, MultiRam fits ALL (even Euro) Apples, can't interfere with slot 1 cards & has no soldered chips!

MultiRam RGB expands to 1 Meg main RAM + 3 Meg's piggyback RAM + 2 Meg's BATTERY BACKED-UP RAM. **MultiRam Iie** expands to 768k & can piggyback w/ MultiRam RGB. **A POSSIBLE 6 MEGABYTES IN ONE SLOT - MORE THAN RAMWORKS II & Flipster™.**

FREE APPLEWORKS EXPANDER SOFTWARE that loads ALL (even printer routines or PARTS of AppleWorks, runs 30 x faster, increased Desktop over 2048k, auto-segments large files onto multiple disks, stores over 23,000 records!) **FREE APPLEWORKS TIME/DATE ON-SCREEN, AUTO-COPY TO RAM, ULTRA-FAST PRODOS/DOS 3.3 RAMDISK & RAMTEST**, optional CP/M & Pascal Ram disk! Printer Buffer free anytime.

	MultiRam RGB Card	MultiRam Iie Card
64k MULTIRAM	169.	129.
128k MULTIRAM	179.	139.
320k MULTIRAM	206.	175.
576k MULTIRAM	241.	214.
832k MULTIRAM	266.	239.
1024k MULTIRAM	284. ← ← ← ← ←	
1344k MULTIRAM	449.	-
1600k MULTIRAM	484.	-
1792k MULTIRAM	519.	-

256k Memory Chips-1 yr warranty (8)	55.
Apple Iie Enhancement Kit	62.
Accelerator Iie-350% speedup card	222.
Accelerator Iie - Pinpoint (special)	249.
Clockworks Card (Thunder Time HC™ comp.)	89.
Pico™ Slimline Drive Iie, Iie, II -	158.
FD-100 Slimline Drive Iie, II -	115.
Pinpoint Program or Spell Checker (ea.)	49.
65C816 EX Co-Processor Card	157.
RGB Monitors, Connectors & Cables*	call

Terms: Add \$4-Ground or \$6-Air shipping & phone # to each U.S. card order (foreign orders/FPO/APO extra). Add 3% for MasterCard/Visa (include #/expir) & P.O.'s (3% 7 Net 30). For fast delivery send Cashier's/Certified check, Money Order, C.O.D. (add \$5) & personal checks accepted (allow 16 days). **Tex res add 6 1/8% tax.**

MultiRam, Ramworks, Ramworks II, Timemaster II, H-O Z ram, Pico, Flipster, respective trademarks of Checkmate Technology, Applied Engineering, WGE, Cinch.

Checkmate Technology's **State-Of-The-Art Iic** cards easily expand your Iic up to 640k, are 100% compatible with all Iic software/hardware, & come with the **SAME FREE SOFTWARE as MULTIRAM Iie (see above)**. **MULTIRAM C** is non-upgradable, **MULTIRAM CX** can be upgraded with a real 65C816 kit (unlike Z-Ram™) to likely run software for the new Apple computer!

- **UNLIKE Z-RAM, THERE ARE NO JUMPER WIRES, CLIPS TO ATTACH, SOLDERED CHIPS, OR DRIVE REMOVAL REQUIRED FOR INSTALLATION.**
- **USES ABOUT 50% LESS POWER** than Z-RAM causing less power supply strain or battery pack drain!
- **15 DAY MONEY BACK SATISFACTION GUARANTEE, 5 YR WARRANTY, & LOWER PRICES** - We sell Iic cards for much less & our software updates are **FREE & AUTOMATIC**, while others charge \$10 or more!

OUR LOWEST PRICE

256k MULTIRAM C	159.
512k MULTIRAM C	189.
256k MULTIRAM CX	189.
512k MULTIRAM CX	237.
65C816 CX Kit (\$10 less w/ card)	129.
VIP Professional w/ any 65C816	117.
1200 Baud Iic Modem w/cable	221.
CP/M & Pascal Ram Disk Iie/Iic (ea)	20.

WHY BUY FROM COIT VALLEY COMPUTERS RATHER THAN SOME MAIL ORDER HOUSES? Only we offer an exclusive 15 day money back satisfaction guarantee, double software, more support, free **automatic** software updates, free 64k with each 256k/512k/768k Iie card. We know the products well, & we have them in stock. **CALL FOR DETAILS, CURRENT PRICES, QUANTITY DISCOUNTS, OR NEW FEATURES! SCHOOLS & GROUPS WELCOME.**

ORDER FORM

COIT VALLEY COMPUTERS (214) 234-5047
14055 Waterfall Way Dallas, Texas 75240

NAME	
ADDRESS	
CITY	STATE ZIP
PHONE (.....)	SIGNATURE A7
QTY DESCRIPTION	PRICE
MC/VISA	SHIPPING
EXP	TOTAL

COIT VALLEY COMPUTERS
(214) 234-5047

14055 Waterfall Way
Dallas, Texas 75240

When I read Bob S-C's article on CRC in the February 1986 AAL, I said, "Very interesting, but who needs it". Well, it wasn't long before I ran into a real need myself!

I bought a used IBM PC-Jr and wanted to put my own routines in an auto-start ROM cartridge. After some sleuthing, I found that the power-up routine checks for signature bytes. If they are present, the routine checks the ROM's CRC, which must be \$0000 or the machine locks up.

Not knowing the 65802 opcodes that Bob used, and being quite familiar with the 8088 language, I decided to translate the PC-Jr's CRC routine from "8088 dis-assembly language" to "plain vanilla 6502-ese". I simulated the 8088's registers with Apple RAM, and wrote subroutines for some of the 16-bit 8088 instructions.

Now here's what I think is strange about CRC's. If you pass all bytes of a set of data through the CRC generator and then the two CRC bytes themselves, the total CRC result is \$0000! The PC-Jr add-on ROMs have the program in all except the last two bytes and the CRC of the program in those last two, so the total CRC for the entire ROM is \$0000.

My 6502 code requires you to enter the start in Apple RAM and the length of the ROM data. For example, for a program starting at \$2000 in Apple RAM, destined to be blown into a 2716 EPROM (2048 bytes), you would enter an address of \$2000 and a length of \$0800. These two values go into the first four bytes of the Apple zero page, so you can use a monitor instruction from inside the S-C Assembler like this:

```
:$00:00 20 00 08
```

My program runs a CRC calculation on all but the last two bytes, and then prints out what the resulting CRC code is. If you store the CRC value in the last two bytes of the ROM image, add two to the length, and re-run my program, the result should be 0000. In a particular example with a 2716, it might look like this:

```
:$00:00 20 00 08      (set up address & length )
:$800G                (run CRC calculation   )
82DF                  (value of CRC computed  )
:$20FE:02 DF          (store CRC in EPROM image)
:$02:02               (increase length by two )
:$800G                (run CRC calculation   )
0000                  (it worked!             )
```

My routines will not win the speed or elegance contests, but they give me the data!

If you want another check on your coding, run a CRC calculation on the Applesoft \$D000 ROM with length \$0800. You should get \$D01E if you have an Apple II+ or original //e version. The enhanced //e gives a CRC of \$3BD4 because of some small changes

Apple made.

By the way, I use my Apple to generate assembly language code for the IBM PC line. I created an 8086/8088 cross assembler based on the S-C Assembler for the purpose. Contact me if you need a tool like this: Don Rindsberg, The Bit Stop, 5958 S. Shenandoah, Mobile, Alabama 36608. Or call at (205) 342-1653.

```

1000 #SAVE ROM CRC CALCULATION
1010 #-----
00- 1020 LOCN .EQ $00,01 ENTER DATA LOCN (L/H)
02- 1030 SIZE .EQ $02,03 ENTER ROM SIZE (L/H)
04- 1040 AL .EQ $04 SIMULATED 8088 REGISTERS
05- 1050 AH .EQ $05
06- 1060 BL .EQ $06
07- 1070 BH .EQ $07
08- 1080 CL .EQ $08
09- 1090 CH .EQ $09
0A- 1100 DL .EQ $0A
0B- 1110 DH .EQ $0B
0C- 1120 PTR .EQ $0C,0D WORK POINTER
0E- 1130 CTR .EQ $0E,0F BYTE COUNTER
1140 #-----
F941- 1150 PRNTAX .EQ $F941
1160 #-----
1170 .OR $300
1180 #-----
0300- A5 00 1190 START LDA LOCN SETUP POINTER
0302- 85 0C 1200 STA PTR TO ROM IMAGE
0304- A5 01 1210 LDA LOCN+1
0306- 85 0D 1220 STA PTR+1
1230 #-----
0308- 38 1240 SEC GET BYTE COUNT - 2
0309- A5 02 1250 LDA SIZE
030B- E9 02 1260 SBC #2
030D- 85 0E 1270 STA CTR
030F- A5 03 1280 LDA SIZE+1
0311- E9 00 1290 SBC #0
0313- 85 0F 1300 STA CTR+1
1310 #-----
0315- A0 FF 1320 LDY #$FF START CRC AT $FFFF
0317- 84 0A 1330 STY DL
0319- 84 0B 1340 STY DH
031B- C8 1350 INY Y=0
031C- 84 05 1360 STY AH INIT AH REG
1370 #-----
031E- B1 0C 1380 .1 LDA (PTR),Y GET NEXT BYTE
0320- 20 3E 03 1390 JSR FOLD.BYTE.INTO.CRC
0323- E6 0C 1400 INC PTR BUMP THE WORK POINTER
0325- D0 02 1410 BNE .2
0327- E6 0D 1420 INC PTR+1
0329- A5 0E 1430 .2 LDA CTR DECREMENT THE BYTE COUNT
032B- D0 02 1440 BNE .3
032D- C6 0F 1450 DEC CTR+1
032F- C6 0E 1460 .3 DEC CTR
0331- A5 0E 1470 LDA CTR TEST IF FINISHED
0333- 05 0F 1480 ORA CTR+1
0335- D0 E7 1490 BNE .1 ...KEEP GOING
0337- A6 0A 1500 LDX DL DISPLAY THE RESULT
0339- A5 0B 1510 LDA DH
033B- 4C 41 F9 1520 JMP PRNTAX
1530 #-----
033E- 45 0B 1540 FOLD.BYTE.INTO.CRC
0340- 85 0B 1550 EOR DH
0342- 85 04 1560 STA DH
0344- 20 6E 03 1570 STA AL
0347- 20 98 03 1580 JSR ROLAX4 8088 "ROL AX,C"
034A- 20 77 03 1590 JSR EORAD 8088 "EOR DX,AX"
034D- A5 0B 1600 JSR ROLAX1 8088 "ROL AX,1"
034F- A6 0A 1610 LDA DH SWAP BYTES IN REG-D
0351- 86 0B 1620 LDX DL
0353- 85 0A 1630 STX DH
0355- 20 98 03 1640 STA DL
0358- 20 83 03 1650 JSR EORAD 8088 "EOR DX,AX"
0358- 20 83 03 1660 JSR RORAX4 8088 "ROR AX,C"

```

```

035B- A5 04      1670      LDA AL
035D- 29 E0      1680      AND #$E0
035F- 85 04      1690      STA AL
0361- 20 98 03   1700      JSR EORAD      8088 "EOR DX,AX"
0364- 20 8C 03   1710      JSR RORAX1    8088 "ROR AX,1"
0367- A5 04      1720      LDA AL
0369- 45 0B      1730      EOR DH
036B- 85 0B      1740      STA DH
036D- 60         1750      RTS
1760      *-----*
1770      * SIMULATE 8088 "ROL AX,C"
1780      *-----*
036E- 20 77 03   1790 ROLAX4 JSR ROLAX1    SHIFT 4 BITS BY SHIFTING
0371- 20 77 03   1800      JSR ROLAX1      1 BIT 4 TIMES
0374- 20 77 03   1810      JSR ROLAX1
1820      *-----*
1830      * SIMULATE 8088 "ROL AX,1"
1840      *-----*
0377- A5 04      1850 ROLAX1 LDA AL      8088 "ROL" SHIFTS END AROUND
0379- 0A         1860      ASL          WITHOUT LEAVING A BIT IN CARRY
037A- 26 05      1870      ROL AH
037C- 90 02      1880      BCC .1      6502 DOES LEAVE A BIT IN CARRY,
037E- 09 01      1890      ORA #$01    SO LETS MERGE CARRY IN HERE.
0380- 85 04      1900      STA AL
0382- 60         1910      RTS
1920      *-----*
1930      * SIMULATE 8088 "ROR AX,C"
1940      *-----*
0383- 20 8C 03   1950 RORAX4 JSR RORAX1    SHIFT 4 BITS BY SHIFTING
0386- 20 8C 03   1960      JSR RORAX1      1 BIT 4 TIMES
0389- 20 8C 03   1970      JSR RORAX1
1980      *-----*
1990      * SIMULATE 8088 "ROR AX,1"
2000      *-----*
038C- A5 05      2010 RORAX1 LDA AH      8088 "ROR" SHIFTS END AROUND
038E- 4A         2020      LSR          WITHOUT LEAVING A BIT IN CARRY
038F- 66 04      2030      ROR AL
0391- 90 02      2040      BCC .1      6502 DOES LEAVE A BIT IN CARRY,
0393- 09 80      2050      ORA #$80    SO LETS MERGE CARRY IN HERE.
0395- 85 05      2060      STA AH
0397- 60         2070      RTS
2080      *-----*
2090      * SIMULATE 8088 "EOR DX,AX"
2100      *-----*
0398- A5 04      2110 EORAD  LDA AL
039A- 45 0A      2120      EOR DL
039C- 85 0A      2130      STA DL
039E- A5 05      2140      LDA AH
03A0- 45 0B      2150      EOR DH
03A2- 85 0B      2160      STA DH
03A4- 60         2170      RTS
2180      *-----*

```

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is \$18 per year in the USA, sent Bulk Mail; add \$3 for First Class postage in USA, Canada, and Mexico; add \$14 postage for other countries. Back issues are available for \$1.80 each (other countries add \$1 per back issue for postage). A subscription to the newsletter and the Monthly Disk containing all source code is \$64 per year in the US, Canada and Mexico, and \$87 to other countries.

All material herein is copyrighted by S-C SOFTWARE CORPORATION, all rights reserved. (Apple is a registered trademark of Apple Computer, Inc.)